Contribution Table

Member	Contribution
Kareem Hesham Sayed	AM: X(t) and m(t) plotting, analytical expressions and practical FT on octave.
Aya Tarek Salem	AM: BW of x(t) and m(t) calculation, LPF implementation with 1 Hz and 0.3 Hz for x(t) + plotting the output of the filter.
Azza Hassan Said	FDM: c1(t) plotting and modulation of x(t) using DSB-SC. Plotting the received messages and the input normalized.
Tayseer Ashraf AbdelHalim	FDM: c2(t) plotting and modulation of m(t) using SSB. Stating whether USB or LSB is used.
Haidy Fathy ElBastawesy	FDM: Appropriate value for c2(t) calculation and plotting of s1(t) + s2(t). Coherent demodulator function.
Mohamed Nasser Mohamed	Digital part 1: Manchester line code function, time and spectral domain graphs.
Abdelrahman Mohamed AbdelMonem	Digital part 1: Polar NRZ line code function, time and spectral domain graphs.
Ahmed Karam AbdelHamid	Digital part 2: Baseband of ASK modulation
Ahmed Haitham Ismail	Digital part 2: Transmission of ASK output
Abdelrahman Mahmoud Attia (Team Leader)	Digital part 2: Receiver with 30,60, and 90 degrees shift.

SPRING 2024

Fundamentals of communication systems (ECE 252s)

Project Report By:

ABDELRAHMAN MAHMOUD ATTIA	2101060
HAIDY FATHY ELBASTAWESY	2101807
TAYSEER ASHRAF ABDELHALIM	2101842
ABDELRAHMAN MOHAMED ABDELMONEM	2100818
AZZA HASSAN SAID	2101808
AHMED KARAM ABDEL-HAMID	2101767
KAREEM HESHAM SAYED	2101780

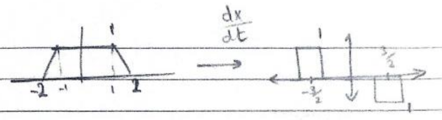
TEAM 2

Content

Analog Communication: AM

Each message signal should be represented in frequency domain using Fourier Transform, since they are non-periodic signals. The analytical solution for each function is shown as following:

- x(t) is trapezium, so it would be solved using differentiation property:
  Zx = 2*sinc(fx).*sin(3*pi*fx)./(2*pi*fx);



$$X(J\omega) = \frac{1}{J\omega}\left(\mathrm{Sinc}f\, e^{\frac{3}{2}J\omega} - \mathrm{Sinc}f\, e^{-\frac{3}{2}J\omega}\right)$$

$$= \mathrm{Sinc}f \cdot \frac{2\sin\left(\frac{3}{2}\cdot 2\pi \cdot f\right)}{2\pi f}$$

$$= 2\,\mathrm{Sinc}f\,\frac{\sin 3\pi f}{2\pi f}$$

- m(t) is a cosine function at time interval [0,6], so it would be solved with the Fourier Transform law:

$$\int_0^6 \cos\omega_0 t\, e^{-J\omega t}$$

$$= \frac{1}{2}\int_0^6 e^{Jt(\omega_0-\omega)} + e^{-Jt(\omega_0+\omega)}$$

$$= \frac{1}{2}\left[\frac{e^{Jt(\omega_0-\omega)}}{J(\omega_0-\omega)} + \frac{e^{-Jt(\omega_0+\omega)}}{-J(\omega_0+\omega)}\right]_0^6$$

$$= \frac{1}{2}\left\{\frac{e^{6J(\omega_0-\omega)}}{J(\omega_0-\omega)} - \frac{e^{-6J(\omega_0+\omega)}}{J(\omega_0+\omega)} - \frac{1}{J(\omega_0-\omega)} + \frac{1}{J(\omega_0+\omega)}\right\}$$

$$= \frac{1}{2J}\left\{\frac{e^{6J(\omega_0-\omega)}-1}{(\omega_0-\omega)} - \frac{e^{-6J(\omega_0+\omega)}-1}{(\omega_0+\omega)}\right\}$$

$$= \frac{1}{2J}\left[\frac{e^{3J(\omega_0-\omega)} - e^{-3J(\omega_0-\omega)}}{e^{-3J(\omega_0-\omega)}(\omega_0-\omega)} - \frac{e^{-3J(\omega_0+\omega)} - e^{3J(\omega_0+\omega)}}{e^{3J(\omega_0+\omega)}(\omega_0+\omega)}\right]$$

$$= \frac{\sin 3(\omega_0-\omega)}{\omega_0-\omega}\, \overset{\to \cos 3(\omega_0-\omega)+J\sin 3(\omega_0-\omega)}{e^{3J(\omega_0-\omega)}} + \frac{\sin 3(\omega_0+\omega)}{\omega_0+\omega}\, \underset{\to \cos 3(\omega_0+\omega)-J\sin\text{-}}{e^{-3J(\omega_0-\omega)}}$$

$$= \frac{1}{2}\left[\frac{\sin 6(\omega_0-\omega)}{\omega_0-\omega} \cancel{+J} - \cancel{J\cos 6(\omega_0-\omega)}\right]$$

$$+ \frac{1}{2}\left[\frac{\sin 6(\omega_0+\omega)}{\omega_0+\omega} \cancel{-J} + \cancel{J\cos 6(\omega_0+\omega)}\right]$$

$$= \frac{1}{2}\left[\frac{\sin 6(\omega_0-\omega)}{\omega_0-\omega} + \frac{\sin 6(\omega_0+\omega)}{\omega_0+\omega}\right]$$

The trapezium function x(t) is transformed to the frequency domain using the fft (Fast Fourier Transform) function, where two low pass filters of frequencies 1KHz and 0.3Hz are applied onto it as shown in figure 1.

It's also apparent upon zooming in the trapezium's frequency spectrum that the analytical and practical Fourier transforms of the signal are almost identical (figure. 2).

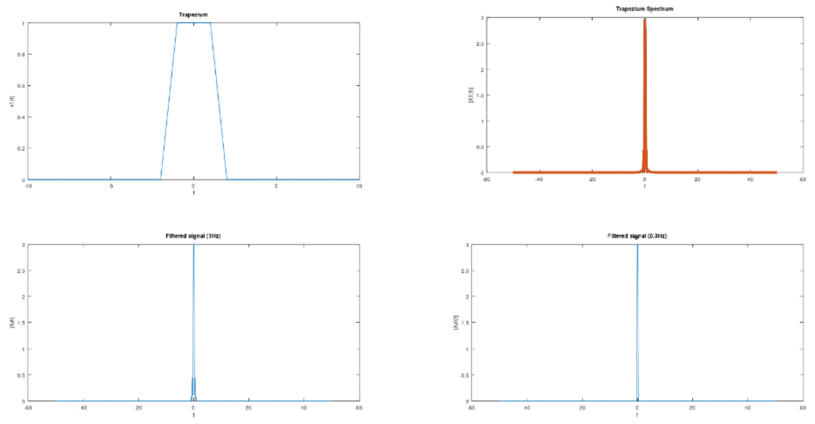


Figure 2 Trapezium signal and LPF implementation.

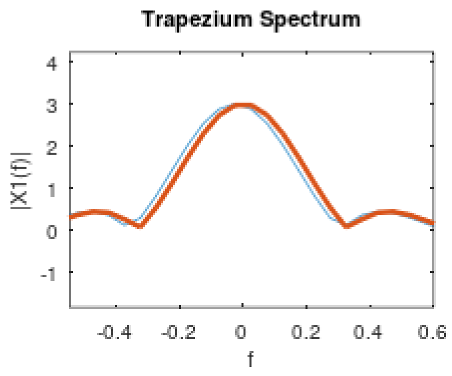


Figure 1: Analytical and Practical Fourier transforms.

The BW of the signal was computed using the following code snippet:

BWx = 0.025 Hz

- For the second signal m(t) the same steps were performed:
  BWm = 3.5232 Hz

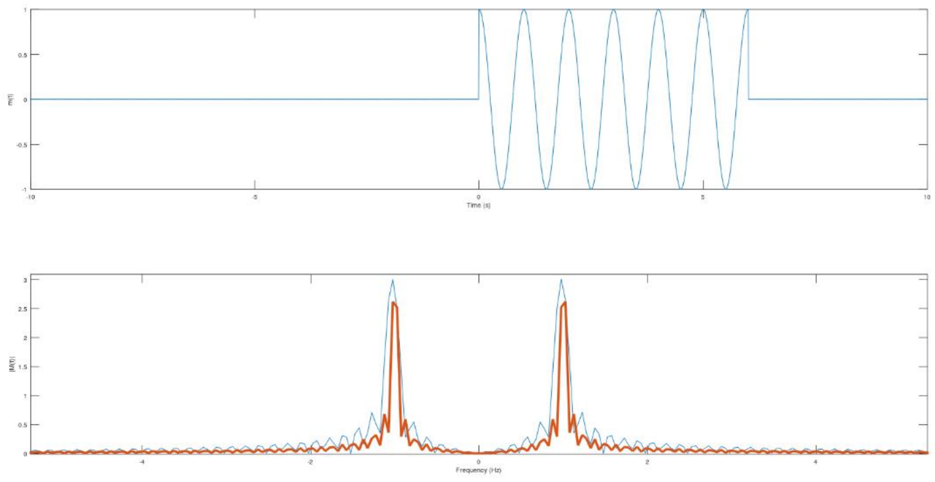


Figure 3 m(t) time and frequency domains. The frequency domain shows the analytical and practical Fourier transform.

AM Octave Code:

```
% Parameters
fs = 100;       % Sampling frequency
df = 0.01;      % Frequency step
ts = 1/fs;      % Sampling interval
fc1 = 20;       % Carrier frequency for x(t)
fc2 = 25.5;     % Carrier frequency for m(t)
T = 1/df; %% Simulation time
Nm = ceil(T/ts); %% Number of time samples

% Define the function x1(t)
t1 = -10 : ts : 10;
x1 = zeros(size(t1));
x1(t1 >= -2 & t1 < -1) = t1(t1 >= -2 & t1 < -1) + 2;
x1(t1 >= -1 & t1 < 1) = 1;
x1(t1 >= 1 & t1 < 2) = -t1(t1 >= 1 & t1 < 2) + 2;

% Calculate the spectrum of x1(t)
X = fftshift(fft(x1)*ts);
Nx = length(X);
fx = (-fs/2:fs/Nx:fs/2 - fs/Nx);
Zx = 2*sinc(fx).*sin(3*pi*fx)./(2*pi*fx);

% Plot x1(t) and its spectrum
figure(1);
subplot(2,2,1);
plot(t1, x1);
xlabel('t');
ylabel('x1(t)');
title('Trapezium');
subplot(2,2,2);
plot(fx, abs(X));
xlabel('f');
ylabel('|X1(f)|');
title('Trapezium Spectrum');
hold on;
plot(fx, abs(Zx), 'LineWidth', 2);

% Calculate power and bandwidth
power_total = sum(abs(X).^2) * df;
power_acc = cumsum(abs(X).^2) * df;
BWx = fx(find(power_acc >= 0.99 * 0.5 * power_total, 1));

% Filter
```

```
Hx = abs(fx) < 1;
Xfilt = X.*Hx;
subplot(2,2,3);
plot(fx, abs(Xfilt));
xlabel('f');
ylabel('|X_filt|');
title('Filtered signal (1Hz)');
Hx2 = abs(fx) < 0.3;
Xfilt2 = X.*Hx2;
subplot(2,2,4);
plot(fx, abs(Xfilt2));
xlabel('f');
ylabel('|X_filt2|');
title('Filtered signal (0.3Hz)');


%% Define function m(t)
m = cos(2*pi*1*t1);
m(t1>6) = 0;
m(t1<0) = 0;
M = fftshift(fft (m))*ts;
Zm = (1/2)*sin(6*((2*pi*fx)+(2*pi)))./((2*pi*fx)+(2*pi)) +
(1/2)*sin(6*((2*pi*fx)-(2*pi)))./((2*pi*fx)-(2*pi));

figure(2);
subplot(2,1,1);
plot(t1,m)
xlabel('Time (s)')
ylabel('m(t)')

subplot(2,1,2);
plot(fx,abs(M))
xlabel('Frequency (Hz)')
ylabel('|M(f)|')
hold on
plot(fx, abs(Zm), 'LineWidth', 2)

%% Calculate BW
Energy_total = sum(abs(M).^2)*df; %% Or  sum(abs(m).^2)*ts;
Index = [~, Index] = min(abs(fx));%% Index of zero frequency
Energy_acc = 0;
for c_index = Index:length(fx)
  Energy_acc = df*abs(M(c_index)).^2 + Energy_acc;
  if(Energy_acc>=0.99*0.5*Energy_total)%% 0.5 because we are calculting
accumulated power in the USB only while the Total power = Power of USB + Power of
LSB
    BWm = fx(c_index);
    break
  end
end
```

Analog Communication: FDM

Modulated signal $s_1(t)$:

The modulated signal is the convolution of the message signal x(t) with the carrier signal c1(t).

The frequency domain of the modulated signal is the multiplication of both signals in frequency domain.

$$S1(f) = X_{filt}(f) \cdot C1(f)$$

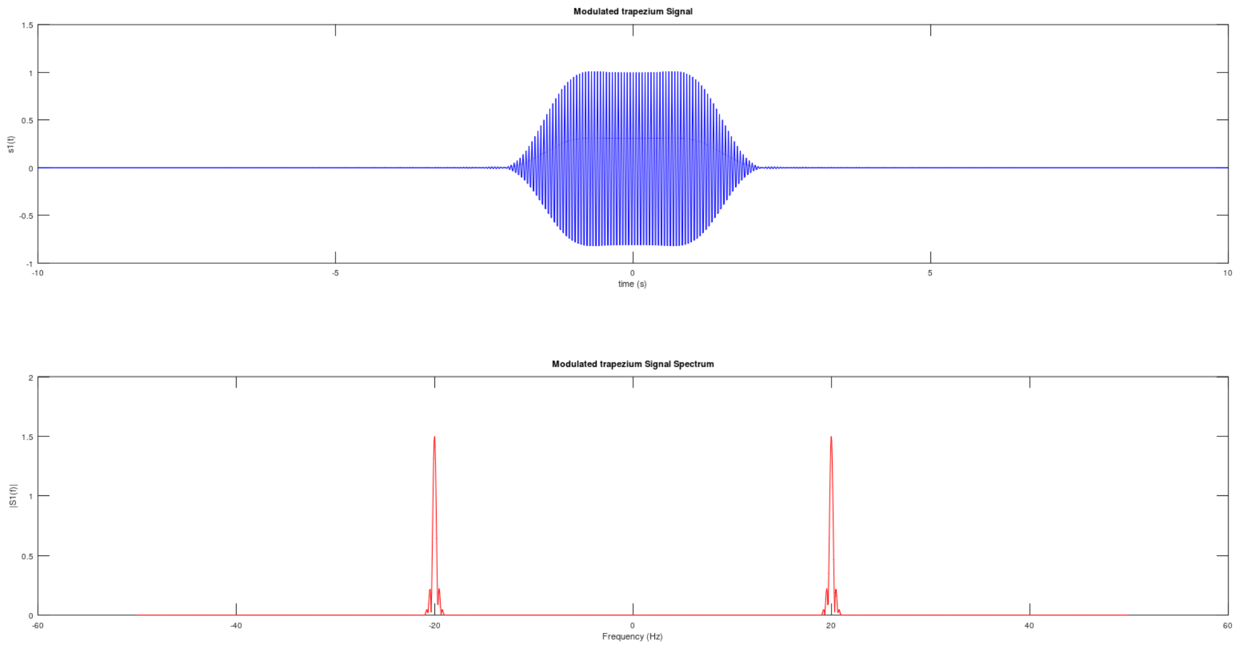


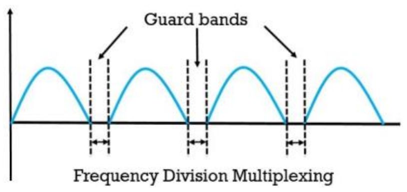
Figure 4. Modulated signal s1 in time and frequency domain.

Modulated signal $s_2(t)$:

The modulated signal is the convolution of the message signal m(t) with the carrier signal c2(t).

To have a guard band of 2.5 Hz with bandwidths 3 Hz, the frequency of second carrier is calculated as the following:

$$f_{c1} + \frac{BW}{2} + 2.5 = f_{c2} - \frac{BW}{2}$$
$$f_{c2} = 20 + 1.5 + 2.5 + 1.5 = \mathbf{25.5}\ \boldsymbol{Hz}$$



Note: SSB bandwidth is located after f_{c2}.

So, the carrier of the signal m(t) is $c_2(t) = \cos(2\pi * 25.5 * t)$.

The frequency domain of the modulated signal is the multiplication of both signals in frequency domain.

$$S2(f) = M(f) \cdot C2(f)$$

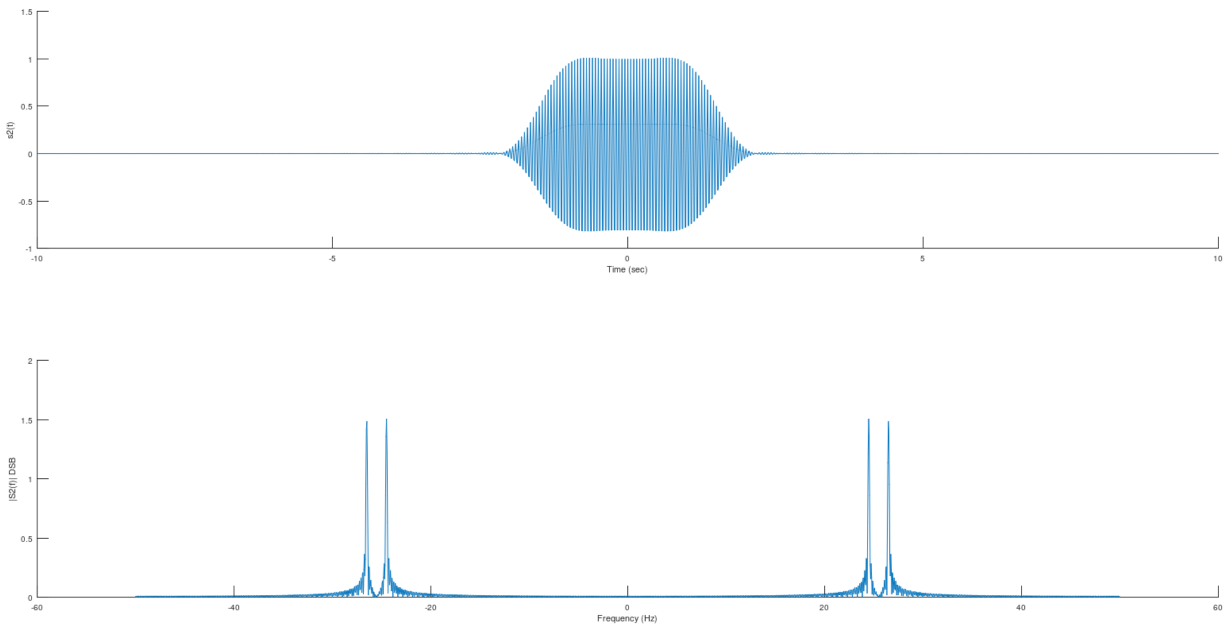


Figure 5. the modulated signal as double-side band in frequency domain.

To get the single-sided band signal, the signal has to pass through a bandpass filter with band from 25.5 Hz till 27 Hz $(f_{c2} + \frac{BW}{2})$ represented in frequency domain as two pulses with width as the band.

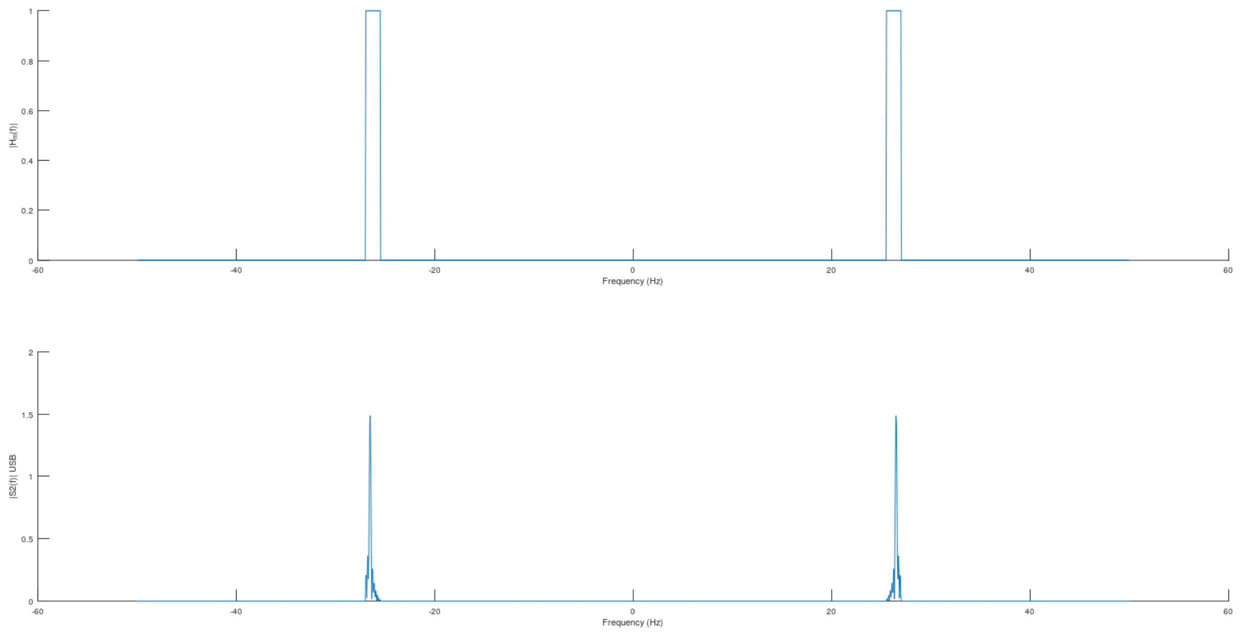


Figure 6. Bandpass filter and modulated SSB signal with upper-side band.

At the receiver, each modulated signal passes through a coherent demodulator consisting of mixer with the carrier of each signal and lowpass filter of bandwidth 1.5 Hz which is half the channel bandwidth.

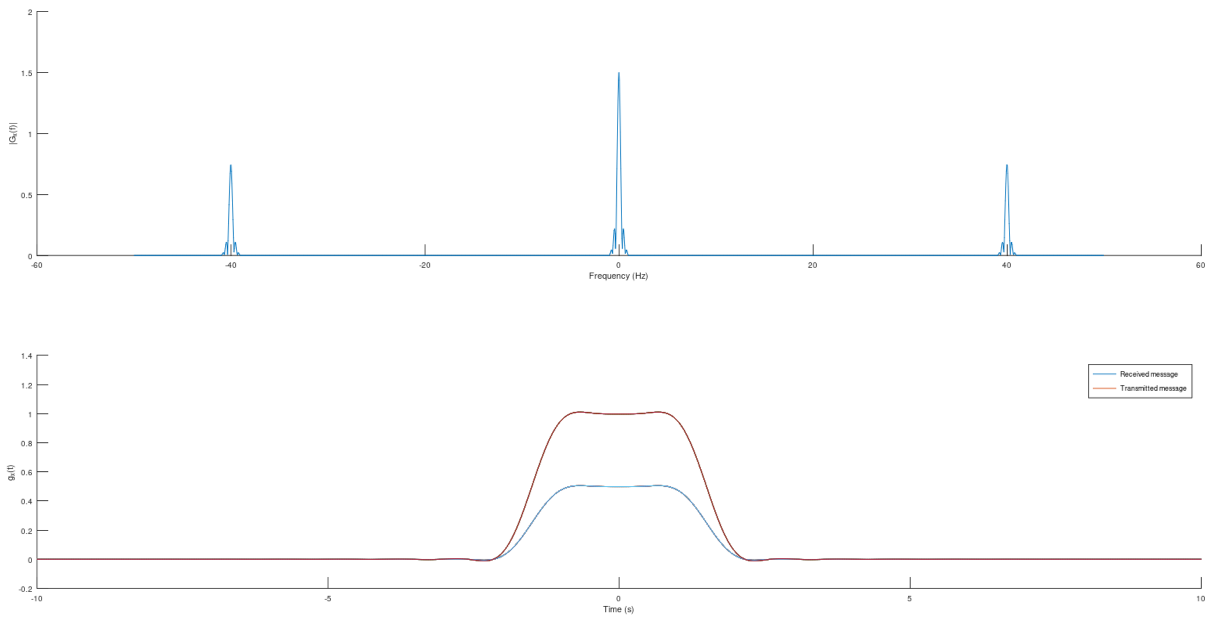


Figure 8. The frequency domain of the received signal X(f) after mixer and before LPF, and the received signal after filter compared to the transmitted signal.

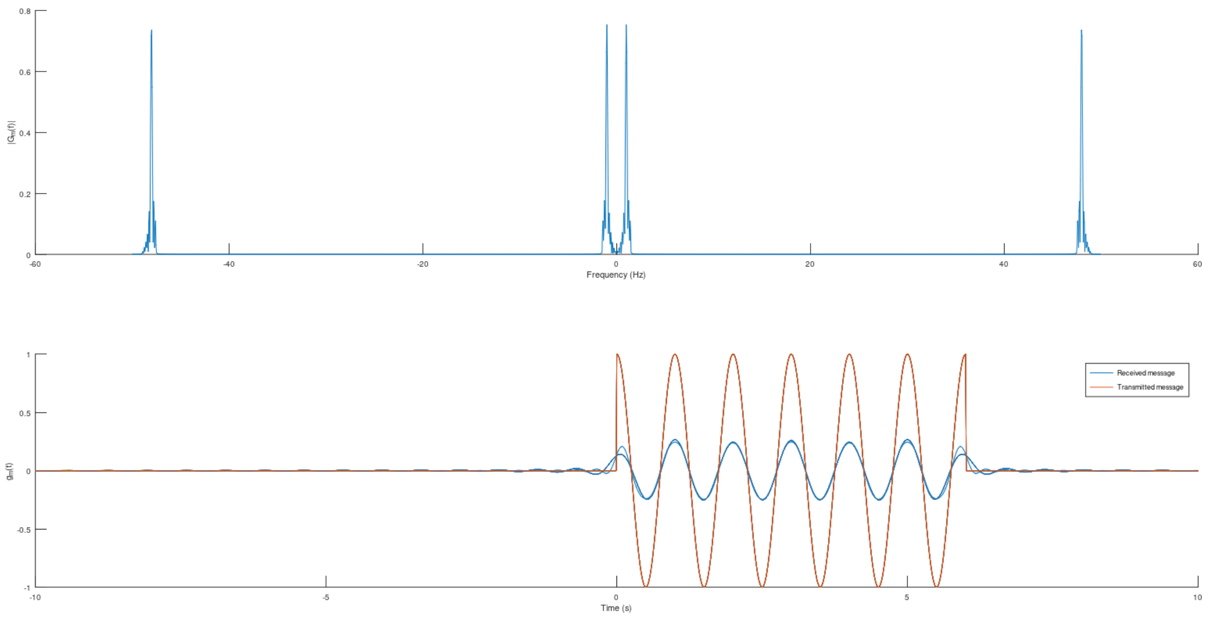
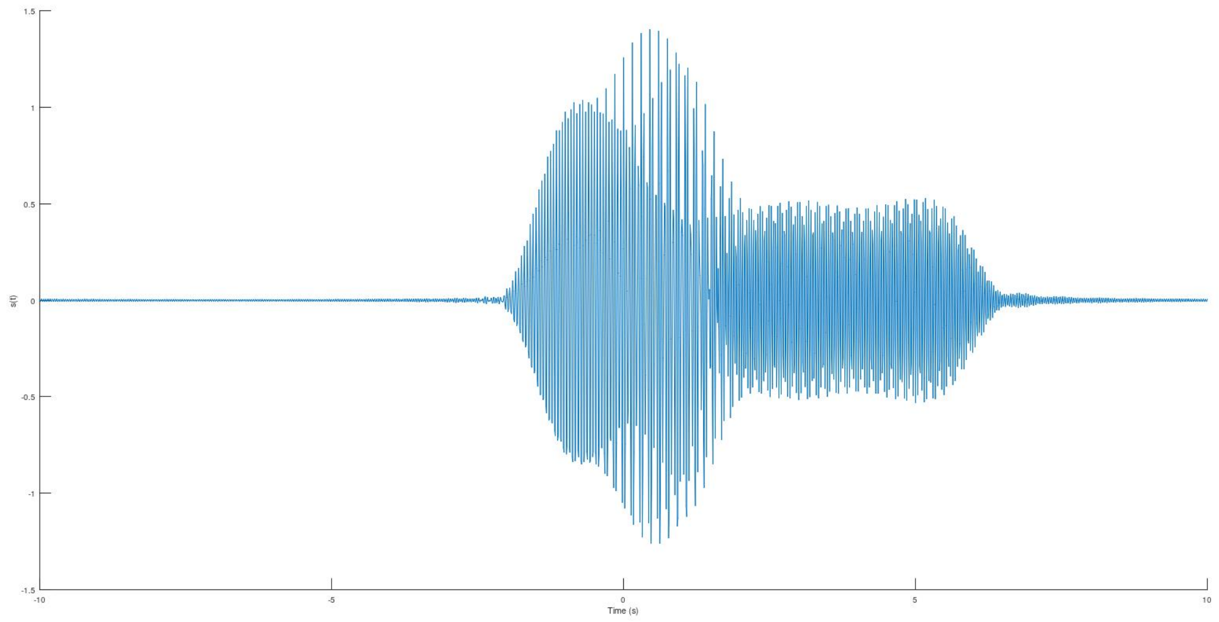
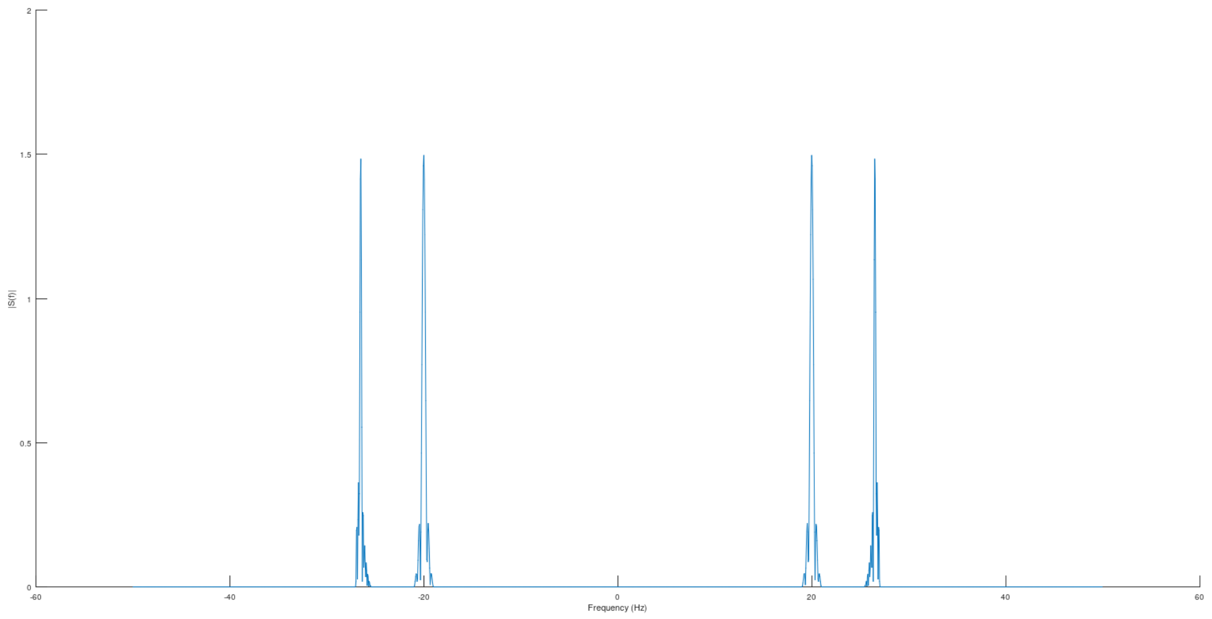


Figure 7. The frequency domain of the received signal M(f) after mixer and before LPF, and the received signal after filter compared to the transmitted signal.

Add the two modulated signals to get a combined signal of both as $s(t) = s_1(t) + s_2(t)$.



Get the frequency domain of the combined signal using Fourier transformation.



FDM Octave Code:

```
%X(t) part
x2 = zeros(size(t1));
x2(t1 >= -2 & t1 < -1) = t1(t1 >= -2 & t1 < -1) + 2;
x2(t1 >= -1 & t1 < 1) = 1;
x2(t1 >= 1 & t1 < 2) = -t1(t1 >= 1 & t1 < 2) + 2;
figure(3)
subplot(2,2,1);
plot(t1,x2);
xlabel('time');
ylabel('x2');

X2 = fftshift(fft(x2)*ts);
subplot(2,2,2);
plot(fx,abs(X2));
xlabel('F');
ylabel('X2');

% Filter
H2filt = abs(fx) < 1;
X2filt = X2.*H2filt;
subplot(2,2,3);
plot(fx,abs(X2filt));
xlabel('F');
ylabel('|X2_filt|');

x2filt=zeros(size(t1));
x2filt=real(ifft(ifftshift(X2filt /ts)));
subplot(2,2,4);
plot(t1,x2filt);
xlabel('time');
ylabel('x2_filt');

% Carrier signal
c1 = cos(2 * pi * fc1 * t1);
C1 = fftshift(fft(c1)) / Nm;

% DSB-SC modulation
s1 = x2filt.*c1;
S1 = fftshift(fft(s1)) *ts;

% Plot modulated trapezium signal and its spectrum
figure(4)
subplot(2,1,1);
```

```
plot(t1, s1, 'b');
xlabel('time (s)');
ylabel('s1(t)');
title('Modulated trapezium Signal');
subplot(2,1,2);
plot(fx, abs(S1), 'r');
xlabel('Frequency (Hz)');
ylabel('|S1(f)|');
title('Modulated trapezium Signal Spectrum');

%%%%%%%%%%%%%%%%%%%%%%%%% Coherent demodulator for X(t)%%%%%%%%%%%%%%%%%%
%% After Mixer
gx = s1.*c1;
Gx = fftshift(fft (gx)) *ts;% 1/N periodic signal, dt non-periodic signal
figure(5)
subplot(2,1,1);
plot(fx,abs(Gx))
xlabel('Frequency (Hz)')
ylabel('|G_x(f)|')
box off

%% LPF (ideal)
H3 = abs(fx)< 1.5;
%% After LPF
x_rec = real(ifft(ifftshift(H3.*Gx) /ts));
subplot(2,1,2);
plot(t1,x_rec)
hold all
plot(t1,x2filt)
legend('Received message','Transmitted message')
xlabel('Time (s)')
ylabel('g_x(t)')
box off

%% m(t) part
%% Carrier
c2 = cos(2*pi*fc2*t1);
C2 = fftshift(fft (c2))*ts;

%% DSB-SC  signal
s2 = m.*c2;
S2 = fftshift(fft (s2)) *ts;% dt non-periodic signal
figure(6)
subplot(2,1,1);
plot(t1, s1)
```

```
xlabel('Time (sec)')
ylabel('s2(t)')
box off
subplot(2,1,2);
plot(fx,abs(S2))
xlabel('Frequency (Hz)')
ylabel('|S2(f)| DSB')
box off

%% SSB signal USB
  %% BPF Filter
  Hm = zeros(size(fx));
  Hm(fx>fc2 & fx<(fc2+1.5))=1;%% +ve frequency
  Hm(fx<-fc2 & fx>-(fc2+1.5))=1;%% +ve frequency
  figure(7)
  subplot(2,1,1);
  plot(fx,Hm)
  xlabel('Frequency (Hz)')
  ylabel('|H_m(f)|')
  box off
  %% Filter spectrum
  S2 = S2.*Hm;
  subplot(2,1,2);
  plot(fx, abs(S2))
  xlabel('Frequency (Hz)')
  ylabel('|S2(f)| USB')
  box off
  %% m_recdulated signal
%% m_recdulated signal
  s2 = real(ifft(ifftshift(S2) / ts));

 %%%%%%%%%%%%%%%%%%%%%%%%% Coherent demodulator for m(t)%%%%%%%%%%%%%%%%%%%
%% Received signal
%% After Mixer
gm = s2.*c2;
Gm = fftshift(fft (gm)) *ts;% 1/N periodic signal, dt non-periodic signal
figure(8);
subplot(2,1,1);
plot(fx,abs(Gm))
xlabel('Frequency (Hz)')
ylabel('|G_m(f)|')
box off

%% LPF (ideal)
Hm = abs(fx)< 1.5;
```

```
%% After LPF
m_rec = real(ifft(ifftshift(Hm.*Gm) /ts));
subplot(2,1,2);
plot(t1,m_rec)
hold all
plot(t1,m)
legend('Received message','Transmitted message')
xlabel('Time (s)')
ylabel('g_m(t)')
box off

% combined signal
s = s1 + s2;
figure(9)
plot(t1, s)
xlabel('Time (s)')
ylabel('s(t)')
box off

S =  fftshift(fft(s)) * ts;
figure(10)
plot(fx, abs(S))
xlabel('Frequency (Hz)')
ylabel('|S(f)|')
box off
```

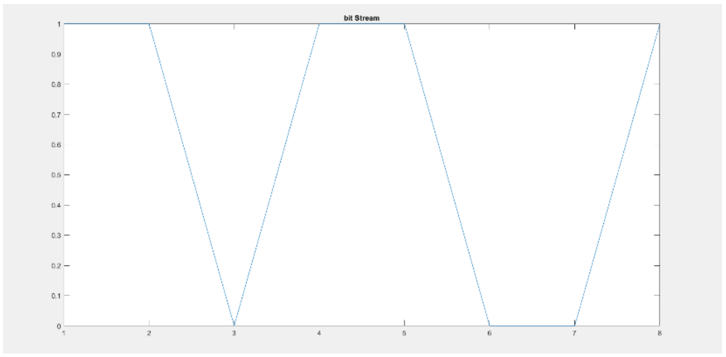
Digital Communication Part 1:
Comparison between PSD OF Manchester and polar non-return to zero line codes

1- 8-bits stream of data

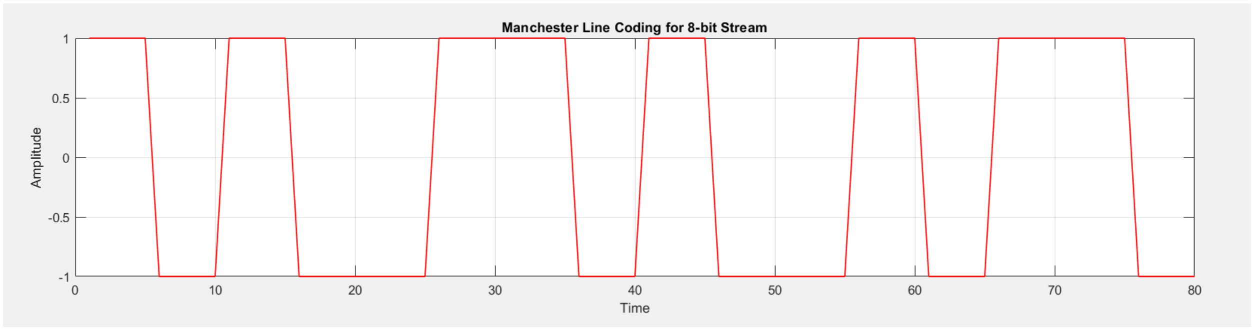
1.1-data pattern

Date = (11 01 10 01)

- Starting with small number of data to make it easy to know if they are correctly coded.

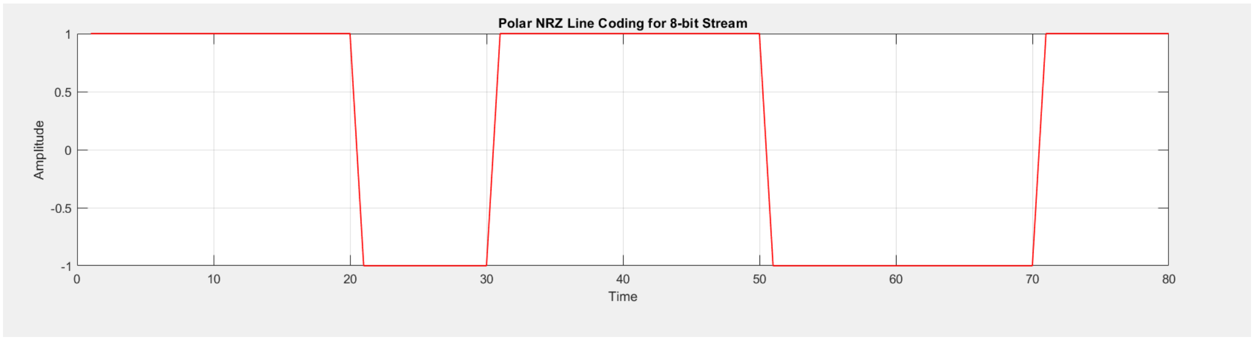


1.2-manchester line code.



- As we see ‘0’ is a transition from low to high, and ‘1’ is a transition from high to low, both transitions happen in Tb/2

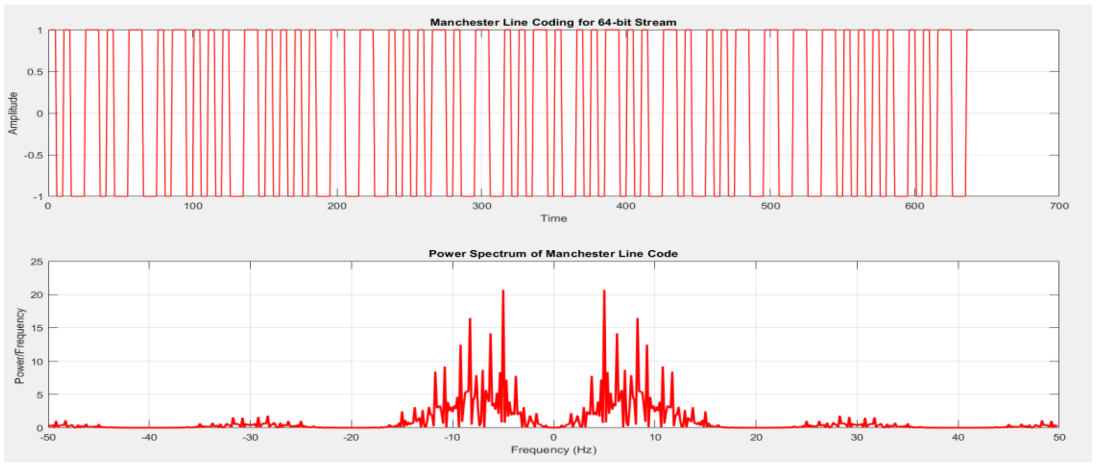
1.3-Polar NRZ lone code.



- More simple in coding ‘1’ is a high level , ‘0’ is a low level of Amp

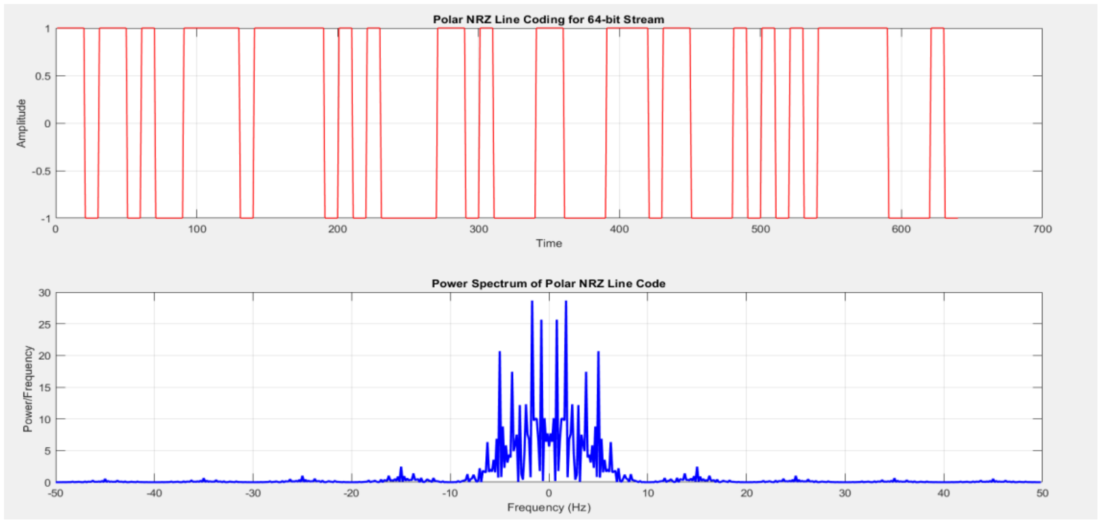
2- 64-bits stream of data

2.1-manchester line code.



- P(f=0) = 0 , no power loss in dc
- Higher BW =2Rb

2.2-Polar NRZ lone code.



- P(f=0) ≠0
- Lower BW = Rb

Digital communication Part 1 Octave code:

```
% Generating 64-bit random stream
bitstream = randi([0,1],1,64);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%Manchester%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Manchester= [];
for i = 1:size(bitstream,2)
    if bitstream(i) == 1
        Manchester =[Manchester [1 1 1 1 -1 -1 -1 -1]];
    elseif bitstream(i) == 0
        Manchester =[Manchester [-1 -1 -1 -1 1 1 1 1]];
    end
end
% Plotting the line code
figure(1);
subplot(2,1,1)
plot(Manchester,'r','LineWidth' ,1);
title('Manchester Line Coding for 64-bit Stream');
xlabel('Time');
ylabel('Amplitude');
grid on;
% Compute the Manchester power spectral density
N = length(Manchester);
fs = 100; % Sampling frequency
df = fs / N; % Frequency resolution
f = (-fs/2) + (0:N-1) * df; % Frequency axis
Manchester_fft = fftshift(fft(Manchester));
psd = abs(Manchester_fft).^2 / N; % Power Spectral Density
% Plotting the power spectrum
subplot(2,1,2)
plot(f,psd, 'r', 'LineWidth', 2);
title('Power Spectrum of Manchester Line Code');
xlabel('Frequency (Hz)');
ylabel('Power/Frequency ');
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Polar NRZ%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Polar_NRZ= [];
for i = 1:size(bitstream,2)
    if bitstream(i) == 1
        Polar_NRZ =[Polar_NRZ [1 1 1 1 1 1 1  1 ]];
    elseif bitstream(i) == 0
        Polar_NRZ =[Polar_NRZ [-1 -1 -1 -1 -1 -1 -1 -1]];
    end
end
```

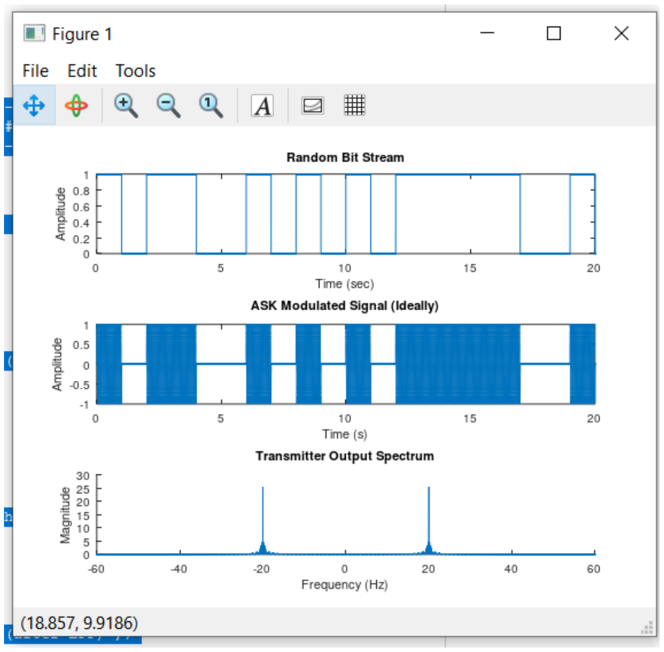
```matlab
% Plotting the Polar NRZ line code
figure(2);
subplot(2,1,1)
plot(Polar_NRZ,'r', 'LineWidth', 1);
title('Polar NRZ Line Coding for 64-bit Stream');
xlabel('Time');
ylabel('Amplitude');
grid on;

% Compute the Polar NRZ power spectral density
N = length(Polar_NRZ);
fs = 100; % Sampling frequency
f = (-fs/2) + (0:N-1) * (fs/N); % Frequency axis
Polar_NRZ_fft = fftshift(fft(Polar_NRZ));
psd_Polar_NRZ = abs(Polar_NRZ_fft).^2 / N; % Power Spectral Density

% Plotting the power spectrum of Polar NRZ
subplot(2,1,2)
plot(f,psd_Polar_NRZ, 'b', 'LineWidth', 2);
title('Power Spectrum of Polar NRZ Line Code');
xlabel('Frequency (Hz)');
ylabel('Power/Frequency ');
grid on;
```

Digital Communication Part 2:
ASK Communication System Simulation Documentation

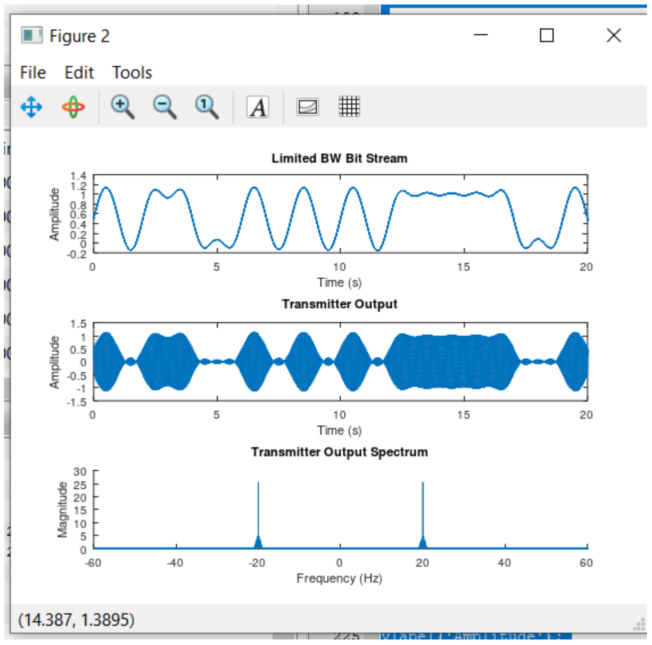
Introduction



This MATLAB script simulates an Amplitude Shift Keying (ASK) communication system. ASK is a digital modulation technique where the amplitude of the carrier signal is varied in accordance with the digital data being transmitted. The script demonstrates the modulation and demodulation processes of ASK, along with the effect of phase shifts on the received signal.

Parameters

- **bit_rate**: Bit rate of the digital data (bits per second).
- **num_bits**: Number of bits to be transmitted.
- **fc**: Carrier frequency of the ASK signal (Hz), which should be higher than the bit rate.
- **ts**: Sampling period of the time domain signal.
- **t**: Time vector representing the duration of the transmission.
- **n**: Length of the time vector.
- **fs**: Sampling frequency.
- **df**: Frequency resolution of the frequency domain signal.



Baseband Data Sequence

- Random binary data sequence (**bits**) is generated with the length specified by **num_bits**.
- The binary data sequence is converted into a baseband signal (**bit_stream**), where each bit is represented by a pulse of specified duration.

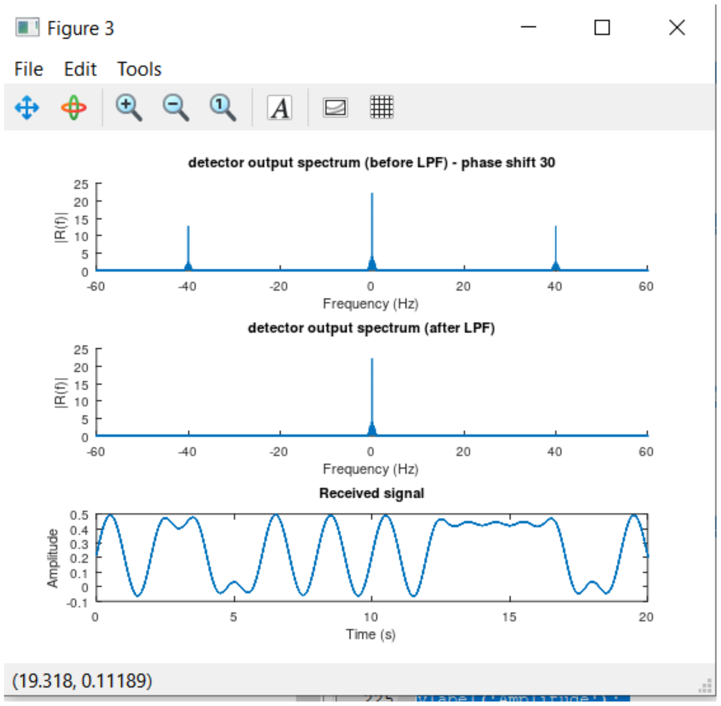
ASK Modulation

- The baseband signal is modulated onto a carrier signal (**c**) using ASK modulation.
- The modulated ASK signal (**ASK_signal**) is obtained by multiplying the baseband signal with the carrier signal.

- The frequency spectrum of the ASK modulated signal (**ASK_signal_f**) is computed using the Fast Fourier Transform (FFT).

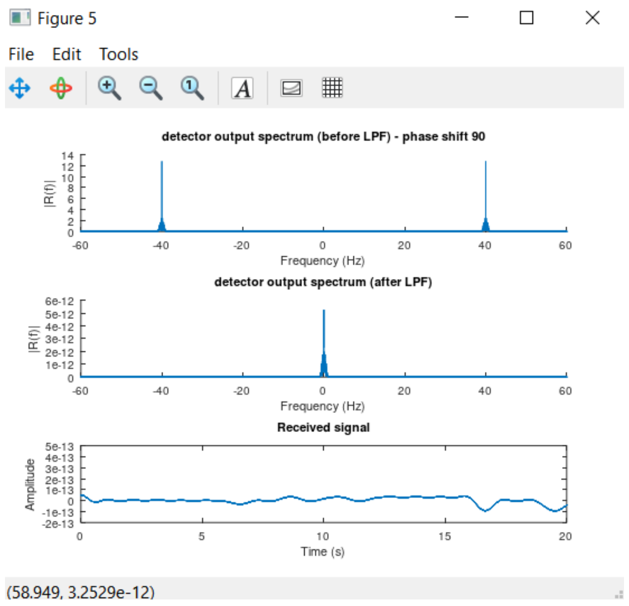
Transmitter Output

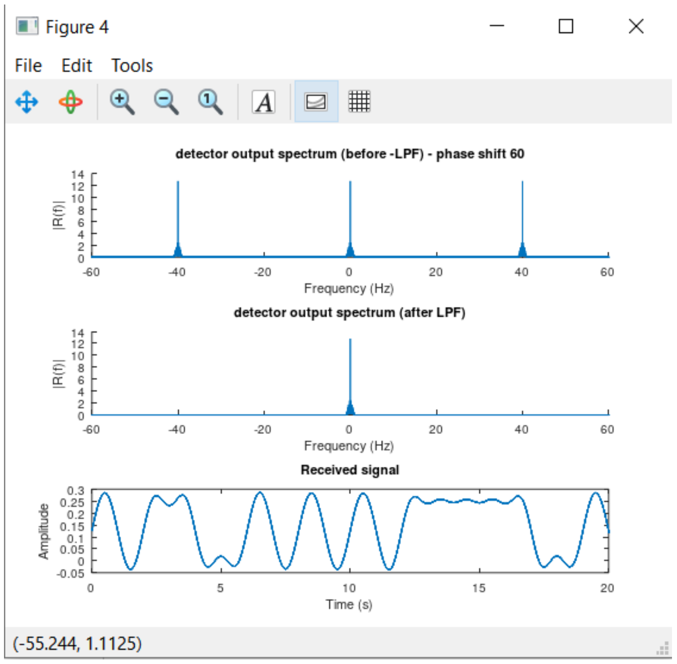
- The modulated ASK signal undergoes filtering to limit the bandwidth using a low-pass filter (**H**).
- The filtered signal (**bit_stream_sent**) represents the transmitter output.
- The transmitter output spectrum (**ASK_signal_sent_f**) is computed and plotted in the frequency domain.



Coherent Receiver

- The transmitted ASK signal is received with a phase shift introduced in the carrier signal.
- Three scenarios are simulated with phase shifts of 30, 60, and 90 degrees.
- Each received signal is demodulated using coherent detection.
- The received signal spectrum before and after filtering is plotted, along with the received signal in the time domain.





Digital communication Part 2 Octave code:

```
#-------------------------------------------------#
# Part2 _ASK
#-------------------------------------------------#
# define Parameters
bit_rate = 1;
num_bits = 100;
fc = 20;           # should be higher than bit_rate
ts = 0.01/fc;
t = 0 : ts : (num_bits/bit_rate)-ts;

n = length(t);
fs = 1/ts;
df = fs/n;

if(rem(n,2)==0) #if n is even
  f = -fs/2:df:fs/2 -df;
else #if n is odd
  f = -fs/2 +df/2:df:fs/2 -df/2;
end

#-------------------------------------------------#
# baseband data sequence
#-------------------------------------------------#


bits = randi([0, 1], 1, num_bits); % out random bits 0s or 1s
bit_stream = zeros(size(t));

for i = 1 : 1 : num_bits
  bit_stream(t >= (i-1)/bit_rate & t <= i/bit_rate) = bits(i);
end

#-------------------------------------------------#
# Base band of  ASK modulated signal
#-------------------------------------------------#

c = sin(2*pi*fc*t);
ASK_signal = bit_stream .* c;   # multiply pulses times the carrier c
ASK_signal_f = fftshift(fft(ASK_signal)) *ts;

figure;
subplot(3, 1, 1);
plot(t, bit_stream);
title('Random Bit Stream');
```

```
xlabel('Time (sec)');
ylabel('Amplitude');
axis([0, 20])

subplot(3, 1, 2);
plot(t, ASK_signal);
title('ASK Modulated Signal (Ideally)');
xlabel('Time (s)');
ylabel('Amplitude');
axis([0, 20])

subplot(3, 1, 3);
plot(f, abs(ASK_signal_f));
title('Transmitter Output Spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
axis([-60, 60])
box off

#------------------------------------------------#
# Transmitter output
#------------------------------------------------#

bit_stream_f = fftshift(fft(bit_stream)) *ts;

H = abs(f) <= bit_rate;   # LPF around Bandwidth (BW = bit rate)
bit_stream_f_sent = H .* bit_stream_f;

bit_stream_sent = ifft(ifftshift(bit_stream_f_sent)/ts);
ASK_signal_sent = bit_stream_sent .* c;
ASK_signal_sent_f = fftshift(fft(ASK_signal_sent)) *ts;

figure;
subplot(3, 1, 1);
plot(t, bit_stream_sent);
title('Limited BW Bit Stream');
xlabel('Time (s)');
ylabel('Amplitude');
axis([0, 20])

subplot(3, 1, 2);
plot(t, ASK_signal_sent);
title('Transmitter Output');
xlabel('Time (s)');
ylabel('Amplitude');
```

```
axis([0, 20])

subplot(3, 1, 3);
plot(f, abs(ASK_signal_sent_f));
title('Transmitter Output Spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
axis([-60, 60])
box off

#--------------------------------------------------#
# Coherent receiver
#--------------------------------------------------#


#--------------------------------------------------#
# receiver with a phase shift 30#
#--------------------------------------------------#


c = sin(2*pi*fc*t + pi/6);
r = ASK_signal_sent .* c;    % Transmitted signal times the carrier
R = fftshift(fft (r)) * ts;  % dt non-periodic signal

figure
subplot(3,1,1)
plot(f,abs(R));
xlabel('Frequency (Hz)');
ylabel('|R(f)|');
title("detector output spectrum (before LPF) - phase shift 30");
axis([-60, 60])
box off

#LPF (ideal)
H = abs(f) <= bit_rate;
#LPF output
H_out = H .* R;
ASK_signal_rec = real(ifft(ifftshift(H_out) /ts));

subplot(3,1,2)
plot(f,abs(H_out));
xlabel('Frequency (Hz)');
ylabel('|R(f)|');
title("detector output spectrum (after LPF)");
axis([-60, 60])
```

```
box off

subplot(3, 1, 3);
plot(t, ASK_signal_rec);
title('Received signal');
xlabel('Time (s)');
ylabel('Amplitude');
axis([0, 20])
#-------------------------------------------------#
# receiver with a phase shift 60#
#-------------------------------------------------#

c = sin(2*pi*fc*t + pi/3);
r = ASK_signal_sent .* c;
R = fftshift(fft (r)) * ts; % dt non-periodic signal

figure
subplot(3,1,1)
plot(f,abs(R));
xlabel('Frequency (Hz)');
ylabel('|R(f)|');
title("detector output spectrum (before -LPF) - phase shift 60");
axis([-60, 60])
box off

#LPF (ideal)
H = abs(f) <= bit_rate;
#LPF output
H_out = H .* R;
ASK_signal_rec = real(ifft(ifftshift(H_out) /ts));

subplot(3,1,2)
plot(f,abs(H_out));
xlabel('Frequency (Hz)');
ylabel('|R(f)|');
title("detector output spectrum (after LPF)");
axis([-60, 60])
box off

subplot(3, 1, 3);
plot(t, ASK_signal_rec);
title('Received signal');
xlabel('Time (s)');
ylabel('Amplitude');
axis([0, 20])
```

```
#-------------------------------------------------#
# receiver with a phase shift 90#
#-------------------------------------------------#

c = sin(2*pi*fc*t + pi/2);
r = ASK_signal_sent .* c;
R = fftshift(fft (r)) * ts; % dt non-periodic signal

figure
subplot(3,1,1)
plot(f,abs(R));
xlabel('Frequency (Hz)');
ylabel('|R(f)|');
title("detector output spectrum (before LPF) - phase shift 90");
axis([-60, 60])
box off

#LPF (ideal)
H = abs(f) <= bit_rate;
#LPF output
H_out = H .* R;
ASK_signal_rec = real(ifft(ifftshift(H_out) /ts));

subplot(3,1,2)
plot(f,abs(H_out));
xlabel('Frequency (Hz)');
ylabel('|R(f)|');
title("detector output spectrum (after LPF)");
axis([-60, 60])
box off

subplot(3, 1, 3);
plot(t, ASK_signal_rec);
title('Received signal');
xlabel('Time (s)');
ylabel('Amplitude');
axis([0, 20])
```

Conclusion

This MATLAB script provides a comprehensive simulation of an ASK communication system, including modulation, filtering, and demodulation processes. It allows for the analysis of the system's performance under different phase shift conditions, providing insights into the impact of phase distortion on the received signal.